Complex Computing in MATLAB and Chebfun

Lloyd N. Trefethen, University of Oxford
December 2019

A work in progress. Please send me corrections and suggestions.

0. Introduction

This booklet has been prepared for a masterlass at the Isaac Newton Institute 9-11 December 2019. It does not give many details, as the main purpose is to collect the code segments to be demonstrated together with a few remarks and references. The focus is mostly on topics from my own work.

In view of the finite time available for both preparation and presentation, just 8 of the 12 sections originally envisaged are included here, and part 3 is barely present at all.

Chebfun, which runs in MATLAB, is freely available from `www.chebfun.org`, and we use both MATLAB and Chebfun without comment. Part 3 depends on additional codes not currently in MATLAB or Chebfun, including `laplace.m`, which is available at `people.maths.ox.ac.uk/trefethen/`.

1. Contour integrals
   - 1.1. Values and coefficients, p. 4
   - 1.2. Zeros, poles, and eigenvalues, p. 12
   - [1.3. Hankel contours]
   - [1.4. Steepest (or not steepest) descent]
2. Polynomial and rational approximation
   - 2.1. Polynomial approximation and Chebfun, p. 18
   - [2.2. Vandermonde with Arnoldi]
   - 2.3. AAA rational approximation, p. 22
   - 2.4. AAA-Lawson minimax approximation, p. 27
3. Harmonic functions and conformal mapping
   - 3.1. Series solutions, p. 29
   - 3.2. AAA compression of computed solutions, p. 29
   - 3.3. Lightning solutions, p. 30
   - [3.4. Numerical uses of conformal maps]

References, p. 31

1. Contour integrals

If a function is analytic in a neighborhood of a domain of interest, then numerical discretizations of contour integrals converge exponentially, i.e., at a rate $O(C^{-n})$ with $C > 1$. Conversely, if a numerical algorithm converges exponentially, there's a good chance that a contour integral offers the best explanation of why.

1.1. Values and coefficients

1.1.1. Evaluating a function via its mean on a circle. Suppose $f(z) = c_0 + c_1 z + c_2 z^2 + \cdots$ is analytic in a neighborhood of the unit disk. Since only the constant term will have a nonzero mean, we see that $f(0) = c_0$ is the mean value of $f(t)$ over the unit circle S:

$$f(0) = \frac{1}{2\pi} \int_S f(t)|dt|. \tag{1}$$

If we discretize S by sampling it in n equispaced points, we get our first numerical method. Here, for example, we evaluate e^0 as the average of e^z over 18 roots of unity:

```
n = 18;
tk = exp(2i*pi*(0:n-1)'/n);
f = @(z) exp(z);
mean(f(tk))
```

1.1.2. Integration via Chebfun or quadrature codes. Of course, one could evaluate the integral in other ways. One could use Chebfun (with or without the `'trig'` flag to make the representation periodic):

```
fc = chebfun(@(x) f(exp(1i*pi*x)),'trig');
mean(fc)
```

Or one could call a MATLAB quadrature code:

```
fm = @(x) f(exp(1i*x));
integral(fm,-pi,pi)/(2*pi)
```

But often we don't need these more general tools.

1.1.3. From mean values to Cauchy integrals. Noting that $|dt| = dt/(it)$ for $t \in S$ turns (1) into a Cauchy integral,

$$f(0) = \frac{1}{2\pi i} \int_S \frac{f(t)}{t}\, dt. \tag{2}$$

Once we have a Cauchy integral, we can deform the contour and still get the same result. For example, here we use Chebfun to integrate over the unit square.

```
side = chebfun('x-1i');
t = join(side, 1i*side, -side, -1i*side);
dtdx = diff(t);
sum((f(t)/t)*dtdx)/(2i*pi)
```

Further examples of contour integrals in Chebfun can be found in the Complex Examples section at `www.chebfun.org`.

1.1.4. Circumventing rounding errors near removable singularities. Integrals over contours are very useful — indeed, without any strong competition that I know of — for evaluating functions where cancellation error may cause trouble near a removable singularity. For example, what's the value of the "phi function" $f(z) = (e^z - 1 - z)/z^2$ at $z = 0$? Direct evaluation with $z \to 0$ fails dramatically, whereas the mean value readily computes full precision:

```
f = @(z) (exp(z)-1-z)./z.^2;
f(1e-5), f(1e-10)
mean(f(tk))
```

An application of this idea to evaluation of radial basis function approximations in the super-exponentially ill-conditioned regime can be found in [20].

1.1.5. The periodic trapezoidal rule. These calculations are examples of the equispaced trapezoidal rule applied to a periodic analytic integrand. The remarkable accuracy of such quadratures was investigated by Poisson in the 1820s, and exponential convergence as $n \to \infty$ was proved by Davis in 1959 [14]. See [TRAP] for a survey.

1.1.6. Derivatives and Taylor coefficients. Replacing f in (2) by its kth derivative gives

$$f^{(k)}(0) = \frac{1}{2\pi i} \int_S \frac{f^{(k)}(t)}{t}\, dt, \tag{3}$$

and if we integrate by parts k times, this becomes the Cauchy integral formula for Taylor coefficients,

$$c_k = \frac{1}{k!} f^{(k)}(0) = \frac{1}{2\pi i} \int_S \frac{f(t)}{t^{k+1}}\, dt. \tag{4}$$

For example, if $f(z) = \sqrt{2 - z}$, we can compute c_4 like this:

```
f = @(z) sqrt(2-z);
n = 40;
tk = exp(2i*pi*(0:n-1)'/n);
mean(f(tk)./tk.^4)
exact = -5/(1024*sqrt(2))
```

1.1.7. Taylor coefficients via FFT. To compute approximations for $c_0, \ldots, c_{n-1}$ all at once from n data values, we could solve an $n \times n$ Vandermonde system of equations or use the explicit formulas of the Discrete Fourier Transform. The work is reduced to $O(n \log n)$ rather than $O(n^2)$ operations if we use the FFT. Here for example is a code segment adapted from `taylor_coeffs` in [TDA].

```
f = @(z) exp(z);
n = 18;
tk = exp(2i*pi*(0:n-1)'/n);
c = real(fft(f(tk))/n)
```

1.1.8. History. Carl Runge in 1904 knew much of this material [44], and an early advocate of complex variables in the computer era was James Lyness [37]. See papers coauthored with Delves (1967) for function values and Moler (1967) and Sande (1971) for higher derivatives. In the words of Lyness and Moler, "Once complex arguments are allowed, the principal difficulties encountered in numerical differentiation simply disappear" [38]. Though the FFT had appeared in 1965, the first few papers do not mention it. An influential review of "Fast Fourier methods in complex analysis" was published by Henrici in 1979 [25].

1.1.9. Aliasing and accuracy. Suppose f is analytic with $|f(z)| \le M$ for $|z| \le R$ for some $R > 1$. Then we can enlarge the contour of (4) to $|z| = R$,

$$c_k = \frac{1}{2\pi i} \int_{|t|=R} \frac{f(t)}{t^{k+1}}\, dt, \tag{5}$$

which leads to the bound known as *Cauchy's estimate,* $|c_k| \le Mr^{-k}$. Now when an integral is evaluated by discretization at the nth roots of unity, the error arises from the fact that on the grid, each power z^k is indistinguishable from its alias $z^{k(\mathrm{mod}\, n)}$. The error in c_k is accordingly equal to $c_{k+n} + c_{k+2n} + \cdots$, so in particular, the error in c_0 is bounded by $MR^{-n}/(1 - R^{-n})$:

$$\text{discretization error in } c_0\text{: } O(R^{-n}). \tag{6}$$

This is exponential convergence as $n \to \infty$, with the rate depending on the size of the disk of analyticity of f. On a computer with rounding errors at the level of $\varepsilon_{\mathrm{mach}}$, since c_0 is computed as a linear combination of terms of order 1, we get

$$\text{rounding error in } c_0\text{: } O(\varepsilon_{\mathrm{mach}}) \tag{7}$$

1.1.10. Choice of radius. It may be advantageous to sample f at equispaced points on the circle $|z| = r$ for some $r < R$. The discretization error in c_k is now dominated by the aliasing of $c_{k+n}z^{k+n}$:

$$\text{discretization error in } c_k\text{: } O((r/R)^{n+k}). \tag{8}$$

This suggests we would like r to be small. On the other hand c_k is computed as a linear combination of values of size $O(r^{-k})$, which leads to

$$\text{rounding error in } c_k\text{: } O(\varepsilon_{\mathrm{mach}} r^{-k}). \tag{9}$$

This suggests we would like r to be large (for $k > 0$). If minimizing the number of samples is not a concern, a good compromise is to pick r large but also use a large value of n. An early paper addressing the choice of radius with an adaptive algorithm was by Fornberg in 1981 [19], and a full analysis has been given by Bornemann [7]. Here is a modification of the code of §1.1.6 to explore different radii. The function has been scaled so that the exact result is 1.

```
f = @(z) sqrt(2-z) / (-5/(1024*sqrt(2)));
n = 8;
while 1
    r = input('r? (e.g 1, 1/2, 1/4,...) ');
    tk = r*exp(2i*pi*(0:n-1)'/n);
    mean(f(tk)./tk.^4)
end
```

1.1.11. Generating functions. One use of Taylor series is as generating functions. For example, the kth Bernoulli number is $k!$ times the kth Taylor coefficient of $z/(e^z - 1)$. Here is a code adapted from `bernoulli` in [TDA]. Note the choice of radius $r = 4$ and the largish value $n = 80$.

```
f = @(z) z./(exp(z)-1);
n = 80; nk = (0:n-1)';
r = 4;
tk = r*exp(2i*pi*nk/n);
c = real(fft(f(tk))).*factorial(nk)./(n*r.^nk);
exact = [1 -1/2 1/6 0 -1/30 0 1/42 0 -1/30 0 5/66 0 -691/2730 ...
         0 7/6 0 -3617/510 0 43867/798 0 -174611/330];
disp('__n__________computed___________exact___')
for k = 0:20
    fprintf('%3d %17.10f %17.10f \n',k,c(k+1),exact(k+1))
end
```

1.1.12. Complex step differentiation. If there were no rounding errors, we would be able to set $r = 10^{-100}$ with $n = 2$ and still get full accuracy in (8). Just this is what happens with the technique of "complex step differentiation" introduced by Squire and Trapp [49]. To compute the derivative of a real function $f(x)$, we perturb x to $x \pm ir$ in the complex plane with a minuscule value of r and then apply the obvious finite difference formula — equivalent to the 2-point trapezoidal rule. For example, here is the derivative of e^x at $x = 0$:

```
f = @(x) exp(x);
r = 1e-100;
deriv = (f(1i*r) - f(-1i*r))/(2i*r)
```

What saves the computation from rounding error troubles is that the real and imaginary parts of f are independently accurate.

1.1.13. Taylor coefficients "below machine precision". Equation (9) raises a fascinating possibility investigated by Bornemann [7]: by sampling on a circle of radius $r > 1$, it should be possible to compute Taylor coefficients at a level below machine precision. This is indeed true, and here, for example, we compute c_{30} for e^z by taking $r = 12$. The reason the accuracy falls short of machine precision is that e^z has magnitude 10^5 on this circle.

```
f = @(z) exp(z)
n = 40;
k = 30;
r = 12;
tk = r*exp(2i*pi*(0:n-1)'/n);
mean(f(tk)./tk.^k)
exact = 1/factorial(k)
```

Note that Taylor coefficients below $\varepsilon_{\text{mach}}$ don't enable us to evaluate f to better accuracy than $\varepsilon_{\text{mach}}$, but they make it possible to evaluate derivatives $f^{(k)}$ to close to that level.

1.1.14. Contour integrals vs. polynomial interpolation. There is an entirely different idea for computing function values and derivatives from discrete samples that we have not yet mentioned: instead of discretizing a contour integral, *compute a polynomial interpolant and then evaluate the interpolant.* Equivalently, *compute FFT approximations to Taylor coefficients and then evaluate the polynomial defined by these coefficients.* The reason this idea is not well known is perhaps that it turns out to be identical to the discretized Cauchy integral for evaluations at the center point $z = 0$. For example, this code segment adapted from `[P*]` of [AKT] evaluates the 20-point polynomial interpolant to e^z at $z = 0.9$ by means of the coefficients:

```
n = 20;
tk = exp(2i*pi*(0:n-1)'/n);
fk = exp(tk);
c = fft(fk)/n;
z = 0.9;
polyval(flipud(c),z)
exact = exp(0.9)
```

Alternatively one can evaluate the polynomial "by values" instead of by coefficients. The standard stable method for this is the barycentric formula [BARY], which takes this form for roots of unity:

```
mean(tk.*fk./(tk-z)) / mean(tk./(tk-z))
```

Note that these results are indeed different from what we get from a discrete Cauchy integral. The formula is

$$f(z) = \frac{1}{2\pi i} \int_S \frac{f(t)}{t - z}\, dt, \tag{10}$$

which on this coarse grid gives a different, far less accurate estimate of $f(0.9)$:

```
mean(tk.*fk./(tk-z))
```

1.1.15. Similarity of barycentric and Cauchy integral formulas. The similarity of the last two displayed lines of code is striking. The barycentric interpolation formula for the approximation to $f(z)$ can be interpreted as the discretized Cauchy integral for $f(z)$ divided by the discretized Cauchy integral for 1! In the words of Helsing and Ojala [24], "One could say that the denominator in this formula compensates for the error in the numerator." This division by 1 is a familiar step in the theory of barycentric formulas, suggesting that the barycentric formula is a "discretized Cauchy integral of the second kind" [AKT]. Ioakimidis et al. [28] interpret the move from integrals to interpolants as "the use of the Cauchy theorem instead of the Cauchy formula."

1.1.16. Accuracy of contour integrals vs. interpolation. The example of §1.1.14 illustrates that a polynomial interpolant may be far more accurate than a discretized Cauchy integral for z near the sample circle. Indeed, an interpolant has error zero at each sample point, whereas a discretized contour integral has error infinity! For $z \neq 0$, eq. (6) worsens from $O(R^{-n})$ to

$$\text{discretization error in } f(z)\text{: } O(\max\{R^{-n}, |z|^n\}), \tag{11}$$

as stated in [AKT], Thm. 2.1. The accuracy of series as opposed to contour integrals near the sample points has been exploited by Helsing and Ojala, cited above, and in the method of QBX, "quadrature by expansion" [31].

1.1.17. Further comparison of contour integrals and interpolants. Contour integrals approximate only within the sample curve, which acts as a cage or filter that projects a problem into a bounded domain [AKT]. Polynomial interpolants, by contrast, converge throughout the disk of analyticity — the phenomenon of "overconvergence" [55]. (Most of their zeros cluster along the circle of analyticity. The prototype of results about this effect is Jentzsch's theorem [29], which applies to Taylor polynomials, and a generalization applicable to our polynomial interpolants was published by Walsh [54].) The localizing nature of discretized contour integrals is related to an advantage they have: for problems involving matrices or rootfinding, as we shall see in §1.2, they may have lower operation counts.

1.1.18. Laurent series on a circle or an annulus. Suppose f is a continuous function defined on the circle S. The Laurent series of f is defined by the same Cauchy integral as in §1.1.6, but now with negative as well as positive powers of z:

$$f(z) = \sum_{k=-\infty}^{\infty} c_k z^k, \quad c_k = \frac{1}{2\pi i} \int_S \frac{f(t)}{t^{k+1}}\, dt. \tag{12}$$

Lipschitz continuity of f on S is enough to ensure convergence of the series there, and if f is analytic, then it converges in the largest annulus $\rho < |z| < R$ in which f is analytic. All the methods presented above have analogues for this case. For example, here is a code as in §1.1.7 to compute Laurent coefficients of $f(z) = \sqrt{1.3 - z}/(z - 1/2)$ from discrete samples in 150 roots of unity.

```
f = @(z) sqrt(1.3-z)./(z-0.5);
n = 150;
tk = exp(2i*pi*(0:n-1)'/n);
c = fft(f(tk))/n;
semilogy(abs(c),'.')
```

1.1.19. Chebyshev analogues. Everything we have discussed involving the origin $z = 0$, monomials z^k, and circles about the origin has analogues involving the interval $[-1, 1]$, Chebyshev polynomials $T_k(x)$, and *Bernstein ellipses,* i.e., ellipses with foci ± 1. This connection is made by the Joukowsky map between z and x,

$$x = (z + z^{-1})/2, \quad T_k(x) = (z^k + z^{-k})/2. \tag{13}$$

This is a 2-to-1 relationship, with each x corresponding to two values z and z^{-1}. If $F(x)$ is defined on $[-1, 1]$, we transplant it to $f(z) = F((z + z^{-1})/2)$ defined on S with $f(z) = f(z^{-1})$. The Laurent coefficients of f satisfy $c_{-k} = c_k$, so we can write

$$f(z) = {\sum_{k=0}^{\infty}}' c_k(z^k + z^{-k}), \quad F(x) = {\sum_{k=0}^{\infty}}' 2c_k T_k(x), \tag{14}$$

where the primes indicate that the terms with $k = 0$ are divided by 2. In other words, the Chebyshev series of F takes the form

$$F(x) = \sum_{k=0}^{\infty} a_k T_k(x), \quad a_k = \begin{cases} c_0 & k = 0, \\ 2c_k & k > 0. \end{cases} \tag{15}$$

For example, here we compute Chebyshev coefficients by sampling on S. Of course, there are alternative ways to do this that exploit the symmetries and avoid the complex arithmetic.

```
F = @(x) exp(x);
f = @(z) F((z+1./z)/2);
n = 32;
tk = exp(2i*pi*(0:n-1)'/n);
c = real(fft(f(tk))/n);
a = [c(1); 2*c(2:n/2)];
exact = [besseli(0,1) 2*besseli(1:n/2-1,1)]';
disp(['________computed___________exact_____'])
disp([a exact])
```

1.1.20. Chebyshev series and Chebfun. Chebfun exploits Chebyshev series for numerical computation with smooth and piecewise smooth functions on intervals and smooth functions in 2D and 3D boxes, disks, balls, and spheres. Here for example is the Chebyshev series of e^x computed in Chebfun.

```
F = chebfun('exp(x)');
plotcoeffs(F)
```

Normally Chebfun does its work with $x \in [-1, 1]$, corresponding to $z \in S$. However, there is a "turbo mode" for computing Chebyshev coefficients by integrals over an adaptively selected larger contour: a Bernstein ellipse in x, corresponding to a circle in z of radius $r > 1$ [56]. This enables the estimation of Chebsyhev coefficients "below machine precision" as in §1.1.13.

```
F = chebfun('exp(x)','turbo');
hold on, plotcoeffs(F,'or'), hold off
```

1.1.21. From scalars to matrices. The discussion so far has been for scalars, but Cauchy integrals work for matrices too via the resolvent integral [27, 30]

$$f(A) = \frac{1}{2\pi i} \int_\Gamma (tI - A)^{-1} f(t) dt, \tag{16}$$

which holds whenever Γ is a closed contour in the region of analyticity of f that encloses the eigenvalues of A. For stable evaluation of matrix functions near removable singularities as in §1.1.4, see [ETD] and [CFEVAL]. As an example of a matrix function, this code adapted from [TDA] computes $A^{1/3}$ for a matrix A with spectrum lying approximately in the disk of radius 1 about $z_0 = 3$. The contour of integration is the circle of radius 2 about that point.

```
N = 10;
A = randn(N)/sqrt(N) + 3*eye(N);
disp('____np_______||B^3 - A||')
for np = 2.^(1:6)
    circle = exp(2i*pi*(1:np)/np);
    z0 = 3; radius = 2;
    z = z0 + radius*circle;
    I = eye(N); B = zeros(N);
    for i = 1:np
        R = inv(z(i)*I-A);
        B = B + R*(z(i)-z0)*z(i)^(1/3);
    end
    B = real(B)/np;
    fprintf('%6d%20.14f\n',np,norm(B^3-A))
end
```

In most applications, one would not compute `inv` as we have done here; the problem is not to compute $f(A)$ explicitly, but to apply it to various vectors. The discretized contour integral reduces this to solutions of linear systems of equations. It tends to be a competitive method when one is dealing with "backslash matrices:" matrices for which solving $Ax = b$ is much more tractable than classical dense operations like computing eigenvalues. For examples of this technique applicable to electronic structure calculations, see [3NICKS] and [35].

1.2. Zeros, poles, and eigenvalues

1.2.1. Zerofinding by polynomial interpolation. Suppose we want to find the zeros of an analytic function f in the unit disk. The simplest method is to interpolate f by a polynomial p at the nth roots of unity, then find the zeros of p by solving a matrix eigenvalue problem with MATLAB `roots`. If f is analytic for $|z| \le R$, the accuracy for simple zeros will be $O(R^{-n})$ ([AKT], Thm. 3.1). Curiously, though the Chebyshev analogue of this "proxy rootfinding" algorithm was proposed by Good in 1961 [23], it is not clear that the unit disk version was ever proposed before [AKT] and [8, sec. 5] in 2014. For example, following `[Pz*]` of [AKT], here are the zeros of $\cos(2\pi z)$ inside S.

```
n = 60;
f = @(z) cos(2*pi*z);
tk = exp(2i*pi*(0:n-1)'/n);
fk = f(tk);
c = fft(fk)/n;
w = roots(flipud(c));
w = w(abs(w)<1)
```

1.2.2. Roots from values instead of coefficients. As usual, there is a dual formulation that computes roots by values rather than by coefficients [10]. Here is the code segment `[Pz]` of [AKT], devised by Alex Townsend.

```
A = [0 fk.'; fk diag(fk)];
B = diag([0; fk./tk]);
w = eig(A,B);
w = w(abs(w)<1)
```

1.2.3. Finding a single zero by a Cauchy integral. Faster zerofinding is possible with contour integrals. Suppose f is analytic in the closed unit disk and is known to be nonzero on S with just a single zero w in the interior. One can compute w by discretizing the formula

$$w = \frac{1}{2\pi i} \int_S \frac{t f'(t)}{f(t)}\, dt, \tag{17}$$

as proposed by McCune [39] and Delves and Lyness [15]. This method requires approximations to $f'(t)$, which can be avoided by using a different integral with a barycentric flavor proposed by Luck and Stevens [36]:

$$w = \int_S \frac{t}{f(t)}\, dt \Big/ \int_S \frac{1}{f(t)}\, dt \tag{18}$$

For example, this code segment adapted from `[LuckStevens]` of [AKT] finds the zero in the disk of $\log(1.1 - z)$. Note that n has to be quite large to get full precision, because of the singularity at $z = 1.1$, but the work is just $O(n)$.

```
n = 300;
f = @(z) log(1.1-z);
tk = exp(2i*pi*(0:n-1)'/n);
fk = f(tk);
w = mean(tk.^2./fk) / mean(tk./fk)
```

1.2.4. Finding K zeros by a Cauchy integral. The algorithm just presented can be generalized to find K zeros in the unit disk by a method proposed by Kravanja and Van Barel in [32, sec. 1.6] and [AKT]. We do not derive the method here but just give the code adapted from `[Cp]` in [AKT]. Again the work is $O(n)$.

```
n = 180;
f = @(z) cos(2*pi*z);
tk = exp(2i*pi*(0:n-1)'/n);
fk = f(tk);
s = ifft(1./fk);
K = 4; KK = 1:K;
H1 = hankel(s(KK+1), s(KK+K));
H2 = hankel(s(KK+2), s(KK+K+1));
w = eig(H2,H1)
```

1.2.5. Counting zeros in the disk. To determine how many zeros f has in the disk, we can use the fact that K is equal to the winding number of f about 0 on S. In terms of Cauchy integrals,

$$K = \frac{1}{2\pi i}\int_S \frac{f'(t)}{f(t)}dt = \frac{1}{2\pi i}\int_S \log(f(t))'dt. \tag{19}$$

Numerical methods for estimating K from this property go back at least to Spira [8, 32]. In Chebfun, we can do this:

```
t = chebfun('exp(1i*pi*x)');
K = sum(diff(f(t))/f(t))/(2i*pi)
```

Or we can avoid integrals and use the Chebfun `unwrap function:`

```
argf = unwrap(angle(f(t)));
plot(argf)
K = (argf(1)-argf(-1))/(2*pi);
title(['K = ' num2str(K)])
```

For a method based on discrete samples rather than continuous functions, we can use the MATLAB `unwrap` command:

```
n = 40;
tk = exp(2i*pi*(0:n-1)'/n);
fk = f(tk);
argf = unwrap(angle(fk([1:end 1])));
plot(argf,'.')
K = (argf(end)-argf(1))/(2*pi);
title(['K = ' num2str(K)])
```

1.2.6. Polefinding by rational interpolation. If f is known to have poles but no zeros in the unit disk, we can find them by finding zeros of $1/f$. This is not a reasonable assumption, however. But there are other methods for finding poles, and again, the most straightforward is based on interpolation. The method proposed in [AKT] is linearized rational interpolation. Given samples at the nth roots of unity of a meromorphic function f known to have K poles in the disk, we determine a rational function $f(z) = p(z)/q(z)$ of type $(n-1-K, K)$ such that $f(z)q(z) = p(z)$ holds at the sample points; this can be done by calling Chebfun `ratinterp`. The roots of q are then approximations to the poles of f in the disk. The choice $n = 40$ in the following code segment is enough to get about 7-digit accuracy; $n = 60$ would give 16 digits.

```
n = 40;
f = @(z) tan(pi*z);
K = 2;
tk = exp(2i*pi*(0:n-1)'/n);
fk = f(tk);
[p,q,rh,mu,nu,poles,res] = ratinterp(fk,n-K-1,K,n,'unitroots');
poles
```

1.2.7. Polefinding by contour integrals. Alternatively, we can find poles by discretized contour integrals using the same algorithm as in §1.2.4, but now applied to $1/f$ instead of f:

```
s = ifft(fk);
KK = 1:K;
H1 = hankel(s(KK+1), s(KK+K));
H2 = hankel(s(KK+2), s(KK+K+1));
w = eig(H2,H1)
```

Note that numbers just produced, with their 7 digits of accuracy, are the same as in §1.2.6. In fact, it is shown in [AKT] that the algorithms of §1.2.6 and §1.2.7 are mathematically equivalent.

1.2.8. Eigenvalues of a matrix. If A is an $N \times N$ matrix, its eigenvalues are the poles of the resolvent function $(zI - A)^{-1}$. This is a matrix function of z, but we can scalarize it by considering

$$f(z) = u^*(zI - A)^{-1}v, \tag{20}$$

where u and v are arbitrary N-vectors. We can now apply one of the polefinding methods above to get approximations to the eigenvalues of A. For an example from [AKT], here we find the two eigenvalues 0.2 and 0.3 in the unit disk of a 4×4 matrix:

```
A =  [3.2  1.5  0.5 -0.5; -1.6  0.0 -0.4  0.6
     -2.1 -2.2  0.2 -0.1; 20.7  9.3  3.9 -3.4];
u = randn(4,1); v = randn(4,1); I = eye(4);
f = @(z) u'*((z*I-A)\v);
n = 100;
tk = exp(2i*pi*(0:n-1)'/n);
fk = arrayfun(f,tk);
s = ifft(fk);
K = 2; KK = 1:K;
H1 = hankel(s(KK+1), s(KK+K));
H2 = hankel(s(KK+2), s(KK+K+1));
w = eig(H2,H1)
```

1.2.9. Projection onto an invariant subspace. All these matrix contour integrals involve an important aspect of spectral projection. To see this in its simplest form, we can take $f(z) = 1$, so that the resolvent integral (16) becomes

$$\frac{1}{2\pi i}\int_\Gamma (tI - A)^{-1}dt. \tag{21}$$

If Γ encloses all the eigenvalues of A, the result must be the identity matrix I. If Γ encloses just some eigenvalues, however, then (21) is the spectral projector P onto the invariant subspace associated with these eigenvalues [30]. Here we compute P for our example matrix A and verify that its eigenvalues are $1, 1, 0, 0$:

```
P = zeros(4);
for k = 1:n
    P = P + tk(k)*inv(tk(k)*I-A);
end
P = P/n;
eig(P)
```

Note that although P is a projector, it is not an orthogonal projector, since A has nonorthogonal eigenvectors. Thus the two nonzero singular values are bigger than 1:

```
svd(P)
```

1.2.10. Projection as a rational filter. The discretization of (21) achieves its projection by multiplying an eigencomponent of a vector v associated with the eigenvalue z by the quantity

$$b(z) = \frac{1}{1 - z^n}, \tag{22}$$

which is related to the Butterworth filter of electrical engineering. Note that $b \approx 1$ for $|z| < 1$ and $b \approx 0$ for $|z| > 1$, as illustrated in this figure from [AKT]:

```
x = linspace(-1.4,1.4,400);
[xx,yy] = meshgrid(x,x); zz = xx + 1i*yy;
n = 32; bb = 1./(1-zz.^n);
aa = abs(bb); aa = min(aa,1.8);
surf(x,x,aa,aa,'edgealpha',0,'facecolor','interp')
colormap(winter), v = colormap; colormap((2+v)/3);
zlim([-3.2 2.9]), camlight, axis equal off
```

1.2.11. Lower-order filters, repeatedly applied. Here now are a pair of figures. On the left, we take $n = 8$ instead of 32 and get a less effective filter than above. On the right, we take $n = 8$ but apply the filter four times, taking the fourth power of (22).

```
n = 8; bb = 1./(1-zz.^n);
subplot(1,2,1)
aa = abs(bb); aa = min(aa,1.8);
surf(x,x,aa,aa,'edgealpha',0,'facecolor','interp')
colormap(winter), v = colormap; colormap((2+v)/3);
zlim([-3.2 2.9]), camlight, axis equal off
subplot(1,2,2)
aa = abs(bb.^4); aa = min(aa,1.8);
surf(x,x,aa,aa,'edgealpha',0,'facecolor','interp')
colormap(winter), v = colormap; colormap((2+v)/3);
zlim([-3.2 2.9]), camlight, axis equal off
```

This powered-up filter with $n = 8$ illustrates an important idea in the application of these ideas to large-scale matrix computations. To suppress certain eigencomponents, you can apply a filter repeatedly and thereby get away with fewer shifts t_k at which a linear system of equations must be solved. This is an important part of the technology in large-scale applications. In the experiment of §1.2.9, for example, if we had used $n = 25$ instead of 100, the smallest eigenvalue of P would have been of order 10^{-4}; but the smallest eigenvalue of P^4 would have been zero to machine precision.

1.2.12. Polizzi ("FEAST") and Sakurai. There is a considerable literature on eigenvalue algorithms based on contour integrals — or rational filters as just described — and their applications in physics. Two main schools of research initiated these methods: Sakurai and his collaborators in Japan [46], and Polizzi and his collaborators in Massachusetts, who gave their software package the name FEAST [42]. Related earlier work was published by Goedecker [22]. For a discussion of the filtering aspects of these methods, see [52].

2. Polynomial and rational approximation

Polynomials are the utterly robust starting point for numerical algorithms in almost every area, as epitomized by Chebfun. Rational functions bring new power in working around poles and other singularities.

2.1. Polynomial approximation and Chebfun

2.1.1. Polynomials in numerical analysis. Most areas of numerical analysis begin with polynomials. For example:

Taylor series. The basis of discretizations in every area.
Interpolation. The starting idea for manipulation of data.
Quadrature. Based on integrating polynomial interpolants.
Approximation theory. Approximation by polynomials is the core topic.
Rootfinding. "Proxy" algorithms first approximate f by p.
Optimization. The prototype is Newton's method: polynomials of degree 2.
Adams and R-K formulas. The starting point is polynomial interpolation.
Spectral methods for differential equations. Based on global polynomials.

2.1.2. Chebyshev series. If f is Lipschitz continuous on $[-1,1]$, it has an absolutely convergent Chebyshev series there,

$$f(x) = \sum_{k=0}^{\infty} a_k T_k(x), \quad a_k = \frac{2}{\pi} \int_{-1}^{1} \frac{f(s)T_k(s)}{\sqrt{1-s^2}} ds, \tag{23}$$

with the factor in front changed from $2/\pi$ to $1/\pi$ for $k = 0$. See §1.1.19 and chapter 3 of [SC ATAP]. For example, here is the Chebyshev series for the Runge function $(1 + 25x^2)^{-1}$.

```
f = chebfun('1/(1+25*x^2)');
plotcoeffs(f)
```

2.1.3. Chebyshev projection and interpolation. From the infinite series with Chebyshev coefficients $\{a_k\}$, one can compute a degree n approximation f_n "by coefficients" by truncation at degree n — the *Chebyshev truncation* or *projection*. Or one can compute an approximation p_n "by values" by interpolation in the Chebyshev points $x_k = \cos(k\pi/n)$, $0 \le k \le n$ — the *Chebyshev interpolant*:

$$f_n(x) = \sum_{k=0}^{n} a_k T_k(x), \quad p_n(x) = \sum_{k=0}^{n} c_k T_k(x). \tag{24}$$

Here are the degree 3 Chebyshev projection and interpolant of $f(x) = |x + 0.1|$:

```
x = chebfun('x');
f = abs(x+0.1);
plot(f,'k'), hold on
n = 3;
fn = chebfun(f,'trunc',n+1);
pn = chebfun(f,n+1);
plot([fn pn])
xk = chebpts(n+1);
plot(xk,f(xk),'.k','markersize',24), hold off
```

The coefficients $\{a_k\}$ and $\{c_k\}$ are related by aliasing formulas analogous to those discussed in §1.1.9 for unit circle ([ATAP], chap. 4).

2.1.4. Stable evaluation of Chebyshev interpolants. The "`polyval(polyfit)`" method of computing Chebyshev interpolants by solving a Vandermonde system of linear equations is exponentially unstable. There are at least three stable alternatives, however: (1) use the barycentric interpolation formula, introduced by Salzer in 1972 [BARY] [47], (2) compute discrete Chebyshev coefficients via FFT, or (3) improve the Vandermonde matrix method with Arnoldi orthogonalization [VANDERMONDE] (§2.2). (This last method is slower for large n, $O(n^3)$ operations.) Chebfun used (1) in its first decade and now uses (2). To illustrate how fast and stable high-degree polynomial interpolation is, here is a degree 10,000 polynomial evaluated and plotted by Chebfun:

```
cheb.gallery('zigzag')
```

2.1.5. Bernstein ellipses and exponential convergence. If f is analytic, it can be analytically continued to a function bounded by $|f(x)| \le M$ in some Bernstein ellipse E_ρ as defined in §1.1.19. By a contour integral going back to Bernstein in 1912 [6], one can show

$$|a_k| \le 2M\rho^{-k}, \quad k \ge 0 \tag{25}$$

([ATAP], Thm. 8.1). By summing up series, we obtain the bounds

$$\|f - f_n\| \le \frac{2M\rho^{-n}}{\rho - 1}, \quad \|f - p_n\| \le \frac{4M\rho^{-n}}{\rho - 1} \tag{26}$$

([ATAP], Thm. 8.2). Chebfun estimates a Bernstein ellipse with `plotregion`:

```
f = chebfun('1/(1+25*x^2)');
plotregion(f)
```

2.1.6. Convergence throughout the Bernstein ellipse. By further arguments based on (25), we find that f_n and p_n both converge exponentially to f for all z in the ellipse, with a rate that slows down as you move out. For example, here is a contour plot showing accuracy of the Chebfun approximation to the Runge function. Blue contours mark errors $10^{-15}, 10^{-12}, \dots, 10^{0}$ and red contours mark errors $10^{3}, 10^{6}, \dots, 10^{15}$.

```
x = linspace(-1.2,1.2,800);
y = linspace(-.6,.6,400);
[xx,yy] = meshgrid(x,y); zz = xx+1i*yy;
f = @(z) 1./(1+25*z.^2);
p = chebfun(f);
ee = log10(abs(f(zz)-p(zz)));
contour(x,y,ee,-15:3:0,'b')
hold on, contour(x,y,ee,3:3:15,'r')
plot([-.2i .2i],'ok','markersize',6), axis equal
```

Outside the Bernstein ellipse of convergence, these polynomial approximations are useless. Here for example we compare the error $f(z)-p(z)$ at $z = 0.1i$, inside the ellipse, and at $z = 0.3i$ outside:

```
f(0.1i) - p(0.1i)
f(0.3i) - p(0.3i)
```

2.1.7. Jentzsch's effect for Chebyshev series. It was mentioned in §1.1.17 that most zeros of Taylor polynomials cluster near the circle of convergence. Analogously, most zeros of Chebyshev interpolants cluster near the the Bernstein ellipse of convergence [54], as illustrated by this code segment for the Runge function.

```
r = roots(p,'all');
plot(r,'.k','markersize',10)
axis equal, hold off
```

These effects show up beautifully in a phase portrait of p, where each pixel is colored by the argument of p there.

```
h = surf(xx,yy,0*zz,angle(-p(zz)));
set(h, 'EdgeColor','none');
caxis([-pi pi]), colormap hsv(600)
view(0,90), axis equal off
```

The master of phase portraits is Elias Wegert, who has used them to animate a textbook [57] and (with Gunter Semmler and others) a series of "Complex Beauties" calendars (`http://www.visual.wegert.com/`).

2.2. Vandermonde with Arnoldi

On the unit circle, the monomials $\{z^k\}$ form well-conditioned bases and all kinds of computations will succeed. On other domains, computations with monomials will often fail because of exponential ill-conditioning. On $[-1, 1]$, this is not a concern since the Chebyshev polynomials offer a well-conditioned alternative. On domains other than disks and intervals, however, computing with polynomials has traditionally been regarded as problematic, and there is an extensive literature of orthogonal polynomials and related ideas for such domains, such as Faber polynomials and Faber–Walsh polynomials [21, 48]. Recently it has been recognized that one can bypass this literature by not working with the Vandermonde matrices per se but instead applying Arnoldi orthogonalization to construct better-behaved matrices with the same column spaces [VANDERMONDE]. See Yuji Nakatsukasa's talk later today.

2.3. AAA rational approximation

2.3.1. Padé approximation. Padé approximation is the generalization of Taylor polynomials to rational functions. Given $m, n \geq 0$ and an analytic function

$$f(z) = a_0 + a_1 z + a_2 z^2 + \cdots \tag{27}$$

defined in a neighborhood of $z = 0$ (or just a formal power series), the type (m, n) Padé approximant to f is the unique rational function of type (m, n) that matches the Taylor series as far as possible (generically, through term $m + n + 1$) [4] [ATAP, chap. 27]. Padé approximants are good at locating poles, and also at approximating function values "beyond" poles — i.e., outside a region of analyticity where a polynomial approximant might be accurate. For example, the function $f(z) = \sqrt{1+z}$ has a disk of analyticity of radius 1. To evaluate $f(3)$, Taylor polynomials are useless, but we can get machine precision with the Padé approximant of type $(16, 16)$:

```
ak = 1; a = ak;
for k = 1:32, ak = ak*(3/2-k)/k; a = [a; ak]; end
taylor_val = sum(a.*3.^(0:32)')
r = padeapprox(a,16,16,0);
pade_val = r(3)
```

2.3.2. Padé approximation and series extrapolation. Padé approximation is the mathematical basis of most of the standard methods for accelerating convergence. Suppose we have a series of numbers

$$V = a_0 + a_1 + a_2 + \cdots \tag{28}$$

that is converging slowly. If we regard $\{a_k\}$ as coefficients of a generating function $f(z)$ as in (27), then our problem is to evaluate $f(1)$, and the radius of convergence of f is typically 1. To do this, we can use Taylor coefficients a_0 through a_{m+n} to construct a Padé approximant r of type (m, n), then evaluate $r(1)$. For example, here are the first 9 coefficients of the function above and the corresponding value $r(1)$ obtained from type $(4, 4)$ Padé approximation:

```
data = a(1:9)
taylor_extrapolant = sum(data)
r = padeapprox(data,4,4);
pade_extrapolant = r(1)
exact = sqrt(2)
```

Aitken extrapolation corresponds to type $(n, 1)$ Padé approximation, and various other methods associated with names like eta, epsilon, Shanks, and Wynn correspond essentially to type (n, n). See [9] and [ATAP], chap. 28.

2.3.3. Approximation on an interval: `minimax, cf, chebpade, ratinterp`. Padé approximations are based at the single point $z = 0$. For rational approximation on $[-1, 1]$, Chebfun offers four choices: minimax approximation (`minimax`, [MINIMAX], [ATAP, chap. 10]), CF (Carathèodory–Fejér) approximation (`cf`, [ATAP, chap. 20]), Chebyshev–Padé approximation (`chebpade`), and linearized rational interpolation or least-squares (`ratinterp`, [GPT], [ATAP, chap. 26]). Here we show type $(6, 6)$ approximants to $f(x) = \tanh(40(x - 0.2))$. The `minimax` error curve equioscillates, and the `cf` approximation is for practical purposes identical to it in this case of a smooth function f. The `ratinterp` approximation would be a true rational interpolant if we called the code with no fourth input argument, but the additional argument 100 here specifies that r should be a linearized least-squares approximant in 100 Chebyshev points.

```
f = chebfun('tanh(40*(x-0.2))');
xx = linspace(-1,1,1000);
subplot(2,2,1)
    [p,q1,r] = minimax(f,6,6);
    plot(xx,f(xx)-r(xx)), axis on, title('minimax')
subplot(2,2,2)
    [p,q2,r] = cf(f,6,6);
    plot(xx,f(xx)-r(xx)), axis on, title('cf')
subplot(2,2,3)
    [p,q3,r] = chebpade(f,6,6);
    plot(xx,f(xx)-r(xx)), axis on, title('chebpade')
subplot(2,2,4)
    [p,q4,r] = ratinterp(f,6,6,100);
    plot(xx,f(xx)-r(xx)), axis on, title('ratinterp')
```

2.3.4. Extrapolation off the interval. One can use a rational approximation on $[-1, 1]$ to estimate quantities elsewhere in the complex plane, and this is a technique that has been applied in investigations of ODEs and PDEs [TEE] [59]. Here we find the poles of the approximations above closest to the pole of f at $0.2 + \pi i/80$.

```
p = 0.2 + 1i*pi/80; pole_exact = p
r = roots(q1,'all'); pole_minimax = r(abs(r-p)==min(abs(r-p)))
r = roots(q2,'all'); pole_cf = r(abs(r-p)==min(abs(r-p)))
r = roots(q3,'all'); pole_chebpade = r(abs(r-p)==min(abs(r-p)))
r = roots(q4,'all'); pole_ratinterp = r(abs(r-p)==min(abs(r-p)))
```

2.3.5. Rational interpolation and least-squares in roots of unity. The Chebfun`ratinterp` command can also be used for rational interpolation or linearized least-squares in roots of unity if the flag `'unitroots'` is specified. An example was given in §1.2.6.

2.3.6. AAA approximation. For type (n, n) rational approximation of almost any data on almost any (discretized) set in the complex plane, a surprisingly fast and reliable new algorithm has recently appeared: AAA approximation, which stands for "adaptive Antoulas–Anderson," pronounced "triple-A" [2] [AAA]. The algorithm represents an approximation in rational barycentric form

$$r(z) = \frac{n(z)}{d(z)} = \sum_{j=0}^{m} \frac{w_j f_j}{z - z_j} \Bigg/ \sum_{j=0}^{m} \frac{w_j}{z - z_j} \,, \tag{29}$$

where $\{z_j\}$ are the *support points* and $\{w_j\}$ are the (barycentric) *weights*. Given a vector Z of N sample points and a coresponding vector of function values F, the algorithm builds up approximations with $m = 0, 1, 2, \ldots$ in a greedy fashion. At each step, first the next support point z_m is selected as a sample point where $|f(z) - r(z)|$ is maximal, and then weights $w_0, \ldots, w_N$ are chosen to minimize the discrete linearized least-squares norm of $f(z)d(z) - n(z)$ over Z. Here, for example, we approximate $f(z) = e^z/(1 + 25z^2)$ on $[-1, 1]$ and plot `errvec`, the maximal error as a function of step number.

```
Z = chebpts(500);
F = exp(Z)./(1+25*Z.^2);
[r,pol,~,~,~,~,errvec] = aaa(F,Z);
err = norm(F-r(Z),inf)
m = length(errvec)-1;
clf, semilogy(0:m,errvec,'.-','markersize',10)
xlabel('x'), ylabel('error')
```

Two of the poles closely match the exact pair $\pm 0.2i$:

```
pol
```

2.3.7. AAA approximation on complex sets. As a complex example, here Z is a collection of random points in a square and in a disk:

```
square = 2*rand(2000,1) + 2i*rand(2000,1) - (1+1i);
disk = square(abs(square)<1);
Z = [square-1.5; disk+1.5];
plot(Z,'.k','markersize',4)
axis equal off, hold on
```

We approximate $f(z) = z \cdot \mathrm{sign}(\mathrm{real}(z))$ on Z:

```
f = @(z) z.*sign(real(z));
[r,pol] = aaa(f(Z),Z);
length(pol)
err = norm(f(Z)-r(Z))
```

Here is a contour plot of the base 10 logarithm of the error.

```
x = linspace(-4,4,400); y = linspace(-2,2,200);
[xx,yy] = meshgrid(x,y); zz = xx + 1i*yy;
ee = log10(abs(f(zz)-r(zz)));
contour(x,y,ee,-14:-1), colorbar
axis on, hold on, ax = axis;
```

2.3.8. AAA poles and branch cuts. The poles of AAA (and other) rational approximations are so interesting that it is rare for AAA aficionados to produce a plot in the complex plane without including the poles. Let's add them to the current plot:

```
plot(pol,'.r','markersize',10)
axis(ax), hold off
```

There has been a great deal of interest in rational approximation theory in the locations of poles of rational approximations. Much of the work on this topic concerns Padé approximations, where a key paper is by Stahl [50]. More recent work, often going beyond Padé, is by Suetin [51]. The essential idea is that the poles of rational approximations line up along branch cuts. Of course these are rational functions, so they can't really have branch cuts. But the poles are doing their best to approximate branch cuts. To see this, let's repeat the same plot but now as a phase portrait as in §2.1.7.

```
h = surf(xx,yy,0*zz,angle(-r(zz)));
set(h, 'EdgeColor','none');
caxis([-pi pi]), colormap hsv(600)
view(0,90), axis(ax), axis equal off
hold on, plot(Z,'.k','markersize',4), hold off
```

Here as another example we approximate $f(z) = \sqrt{1-z^2}$ on a rectangle. The singularities lie at the bottom corners.

```
subplot(1,2,1)
s = tanh(linspace(-12,12,1500)');
Z = [s; 1+.5i+.5i*s; 1i-s; -1+.5i-.5i*s];
F = sqrt(1-Z.^2);
[r,pol,~,~,~,~,~,errvec] = aaa(F,Z,'tol',1e-8);
plot(pol,'.r','markersize',10)
axis([-2 2 -2.5 1.5]), axis square, hold on
hold on, plot(Z,'-k')

subplot(1,2,2)
x = linspace(-2,2,200); y = linspace(-2.5,1.5,200);
[xx,yy] = meshgrid(x,y); zz = xx + 1i*yy;
h = surf(xx,yy,0*zz,angle(-r(zz)));
axis([-2 2 -2.5 1.5]), axis square, hold on
set(h, 'EdgeColor','none');
caxis([-pi pi]), colormap hsv(600), view(0,90)
hold on, plot(Z,'-k')
```

2.3.9. Root-exponential convergence for functions with singularities. When there is a branch point singularity on the boundary of an approximation domain, as in these examples, the convergence will typically be root-exponential, i.e., at a rate $O(C^{-\sqrt{n}})$ with $C > 1$. Here for example we plot the convergence of the AAA approximation above, with not m but $\sqrt{m}$ on the abscissa. The approximately straight data illustrates the root-exponential effect. (Note that to see this properly, we needed to use a very strongly clustered grid Z near the singularities, achieved with the tanh function. With a less clustered grid, the discrete sample set fails to approximate the continuum effectively.)

```
clf
m = length(errvec) - 1;
semilogy(sqrt(0:m),errvec,'.-','markersize',10)
```

The ability of rational functions to achieve root-exponential convergence near singularities was discovered by Newman in 1964 for approximation of $|x|$ on $[-1, 1]$ [41]; see [ATAP], chap. 25. In [LIGHTNING] it is proved that the effect generalizes to a wide class of functions with boundary singularities. The mathematical ideas involved are those of potential theory [34, 45, 50, 51].

2.3.10. Exponential clustering of poles and zeros. To achieve root-exponential convergence near singularities, rational approximations cluster poles and zeros exponentially near there. For example, here we plot the distances of the poles just computed from the singularity at $z = 1$:

```
d = sort(abs(pol-1));
d = d(d<1);
semilogy(d,'.','markersize',14)
```

The straight line shows extremely regular exponential clustering until the last few data points, which are a little more closely clustered still. Remember, AAA has not been instructed to put the poles here; this is just a corollary of its construction of a good approximation. What's going on is a kind of multiscale approximation, with poles at different distance from the singularity having their main influences on the corresponding length scales. The reason the rate is root-exponential rather than exponential is that there is a balance between global resolution, determined by spacing between poles, and local resolution, determined by closeness of poles to the singularity. Such balances are familiar to experts in adaptive *h*-*p* finite element methods.

2.4. AAA-Lawson minimax approximation

2.4.1. Barycentric representations in "alpha-beta mode." The barycentric representation (29) forces r to interpolate the data $\{f_j\}$ at the support points $\{z_j\}$, assuming the weights $\{w_j\}$ are nonzero. This gives AAA approximation much of its power, but it also prevents the approximants from being arbitrary rational functions. In particular, though AAA approximants may be near-minimax, they cannot be truly minimax. For that, we need $2m+1$ rather than just $m+1$ degrees of freedom, and this can be provided by working with a barycentric representation in "alpha-beta mode:"

$$r(z) = \frac{n(z)}{d(z)} = \sum_{j=0}^{m} \frac{\alpha_j}{z - z_j} \Bigg/ \sum_{j=0}^{m} \frac{\beta_j}{z - z_j} \,. \tag{30}$$

As $\{\alpha_j\}$ and $\{\beta_j\}$ range over all complex values with at least one β_j nonzero, (30) ranges over all type (m,m) rational functions [LAWSON, Thm. 3.1].

2.4.2. AAA-Lawson approximation. The AAA-Lawson algorithm, introduced in [LAWSON], assumes that a degree m has been specified; the aim is to find the type (m,m) minimax approximation to data F on a discrete set Z. (Uniqueness is not guaranteed in principle, though this doesn't seem to be an issue in practice [53].) There is little literature on this problem, the main algorithm being by Ellacott and Williams in 1976 [11]. First, the AAA-Lawson algorithm takes m steps of standard AAA, obtaining thereby a set of support points $z_0, \dots, z_m$ that are well adapted to F. Then a phase of *iteratively reweighted least-squares approximation* is entered, running by default for 20 steps. In this phase, the weights α_j and β_j are adjusted at eash step to drive a linear least-squares problem toward the minimax solution, an idea introduced for more standard (and usually linear) minimax problems by Lawson in 1961 [33, 43]. To illustrate, we execute the code segments from the help text of the Chebfun `aaa` code. First we compute type $(15,15)$ approximations of $\tanh(20x)$ on $[-1,1]$ with both AAA and AAA-Lawson.

```
X = linspace(-1,1,1000);
F = tanh(20*X);
r = aaa(F,X,'degree',15,'lawson',0);
plot(X,F-r(X)), hold on
r = aaa(F,X,'degree',15);
plot(X,F-r(X)), hold off
```

Next we compute type $(3,3)$ approximations of e^z on the unit circle. The minimax error curve, with winding number 7 as could be confirmed by the methods of §1.2.5, is not a perfect circle, but it approximates one extremely closely, a phenomenon identified in [NEARCIRC] and [RATCF].

```
Z = exp(1i*pi*linspace(-1,1,1000));
G = exp(Z);
r = aaa(G,Z,'degree',3,'lawson',0);
plot(G-r(Z)), axis equal, hold on
r = aaa(G,Z,'degree',3);
plot(G-r(Z)), hold off
```

3. Harmonic functions and conformal mapping

Conformal mapping is tightly tied to Laplace problems, and here we emphasize algorithms that apply equally to both. Our theme is the use of expansions with unknown coefficients determined by solving least-squares problems.

3.1. Series solutions

Many Laplace problems can be solved by least-squares approximation on the boundary. The starting idea is to represent u as the real part of a polynomial with coefficients determined by least-squares. See [12], [TDA], and [SERIES].

The conformal map of a simply-connection region to a disk can be reduced to a Laplace problem. This is an old idea pursued most recently in [RATCONFORMAL, sec. 2] and implemented in the Chebfun `conformal` command when called with the `'poly'` flag. Three demonstrations are given in the `conformal` help text: the "blade" and a rectangle. The latter is nonsingular at the corners, so a polynomial representation is successful.

```
C = chebfun('2*cos(t)+1i*(sin(t)+2*cos(t).^3)',[0 2*pi],'trig');
conformal(C, 'poly', 'plots');

s = chebfun('s');
C = join(s-.5i,1+.5i*s,.5i-s,-1-.5i*s);
conformal(C, 'poly', 'plots');
```

Such computations depend crucially on the use of Vandermonde with Arnoldi, as described in §2.2.

3.2. AAA compression of computed solutions

AAA approximation applies beautifully to conformal maps. If the boundary correspondence function has been computed that relates the discretized boundaries Z and W of two domains, then `r = aaa(W,Z)` may be all that is needed to construct a rational representation of the result. The reverse command `rinv = aaa(Z,W)` gives an analogous representation of the inverse map. The speedups involved in this method are typically factors of 10–1000. These ideas were introduced in [AAACONFORMAL] with applications in particular to Driscoll's Schwarz-Christoffel Toolbox for MATLAB [17] and to computations with the Kerzman-Stein integral equation, such as these suggested in Chebfun `conformal`, which is now called without the `'poly'` flag.

```
C = chebfun('exp(pi*1i*t)*(1+.3*cos(6*pi*t))','trig');
conformal(C, 'plots', 'tol', 1e-8, 'numbers');

C = chebfun('exp(pi*1i*t)','trig')*(1+.15*randnfun(.2,'trig'));
[f, finv] = conformal(C, 'plots');
```

Further examples of AAA compression of conformal maps and their inverses are given in [RATCONFORMAL], including a doubly connected region treated by Laurent polynomials and a multiply-connected region treated with Nasser's code PlgCirMap [40].

3.3. Lightning solutions

"Lightning Laplace solvers" were introduced in [LIGHTNING]. The idea is to take advantage of the root-exponential clustering effects described in §§2.3.9–2.3.10. The solution u to a Laplace problem is represented as the real part of a rational function that combines a smooth polynoimal "Runge term" with a rational "Newman term" with clustered poles. The poles are placed a priori, making the computation linear and robust. Root-exponential convergence is proved in theory in [LIGHTNING], and theory meets practice thanks to the use of Vandermonde with Arnoldi.

The name "lightning Laplace" refers not just to the speed of this method, but to the underlying idea, which takes advantage of the same mathematics that makes lightning strike at sharp corners.

For demonstrations we can run the examples of the help text of `laplace.m` at `people.maths.ox.ac.uk/trefethen/`. Here is the NA Digest problem of November 2018:

```
exact = 1.0267919261073
[u,maxerr] = laplace('L','plots','tol',1e-10);
u(.99+.99i), maxerr
```

Here are some other examples.

```
laplace([0 1 1+1i 1i],[0 0 0 1],'plots');

laplace('iso',1:8,'plots');

laplace(8,@(z) exp(real(z)),'plots');

laplace(-8,@(z) exp(real(z)),'plots');

laplace({[1 .5] 1+1i -1+1i -1},'plots');

laplace('circleL',[0 1 0 0 0],'plots');
```

Applications of `laplace` to the computation of conformal maps are described in [`ratconformal`] and can be illustrated with examples like these.

```
P = [-1-1i 1-1i 1+.2i .2+.2i .2+1i -1+1i];
conformalR(P,'plots');

conformalR('iso','plots');

conformalR(8,'plots');

conformalR(-8,'plots');
```

The lightning Laplace idea can be extended to the Helmholtz [HELMHOLTZ] and biharmonic equations (work of Pablo Brubeck, not yet published).

Trefethen references

[CHOPPING] J. Aurentz and L. N. Trefethen, *Chopping a Chebyshev series,* ACM Trans. Math. Softw., 43 (2017), p. 33.

[AKT] A. P. Austin, P. Kravanja, and L. N. Trefethen, *Numerical algorithms based on analytic function values at roots of unity,* SIAM J. Numer. Anal., 52 (2014), pp. 1795–1821.

[BARY] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation,* SIAM Rev., 46 (2004), pp. 501–517.

[VANDERMONDE] P. D. Brubeck, Y. Nakatsukasa, and L. N. Trefethen, *Vandermonde with Arnoldi,* SIAM Rev., submitted.

[FARADAY] S. J. Chapman, D. P. Hewett, and L. N. Trefethen, *Mathematics of the Faraday cage,* SIAM Rev., 57 (2015), pp. 398–417.

[SC] T. A. Driscoll and L. N. Trefethen, *Schwarz–Christoffel Mapping,* Cambridge, 2002.

[AAACONFORMAL] A. Gopal and L. N. Trefethen, *Representation of conformal maps by rational functions,* Numer. Math., 142 (2019), pp. 359–382.

[HELMHOLTZ] A. Gopal and L. N. Trefethen, *New Laplace and Helmholtz solvers,* Proc. Nat. Acad. Sci. USA 116 (2019), p. 10223.

[LIGHTNING] A. Gopal and L. N. Trefethen, *Solving Laplace problems with corner singularities via rational functions,* SIAM J. Numer. Anal., 57 (2019), pp. 2074–2094.

[CHEBFUN] T. A. Driscoll, N. Hale, and L. N. Trefethen, Chebfun Guide, Pafnuty Press, 2014; see also `www.chebfun.org`.

[MINIMAX] S.-I. Filip, Y. Nakatsukasa, L. N. Trefethen, and B. Beckermann, *Rational minimax approximation via adaptive barycentric representations,* SIAM J. Sci. Comput., 40 (2018), pp. A2427–A2455.

[GPT] P. Gonnet, R. Pachón, and L. N. Trefethen, *Robust rational interpolation and least-squares,* Elect. Trans. Numer. Anal., 38 (2011), pp. 146–167.

[3NICKS] N. Hale, N. J. Higham, and L. N. Trefethen, *Computing A^α, $\log(A)$, and related matrix functions by contour integrals,* SIAM J. Numer. Anal., 46 (2008), pp. 2505–2523.

[ETD] A.-K. Kassam and L. N. Trefethen, *Fourth-order time-stepping for stiff PDEs,* SIAM J. Sci. Comput., 26 (2005), pp. 1214–1233.

[AAA] Y. Nakatsukasa, O. Sète, and L. N. Trefethen, *The AAA algorithm for rational approximation,* SIAM J. Sci. Comp., 40 (2018), pp. A1494–A1522.

[LAWSON] Y. Nakatsukasa and L. N. Trefethen, *An algorithm for real and complex rational minimax approximation,* SIAM J. Sci. Comp., submitted.

[CFEVAL] T. Schmelzer and L. N. Trefethen, *Evaluating matrix functions for exponential integrators via Carathéodory–Fejér approximation and contour integrals,* Elect. Trans. Numer. Anal., 29 (2007), pp. 1–18.

[TEE] W. Tee and L. N. Trefethen, *A rational spectral collocation method with adaptively transformed Chebyshev grid ponits,* SIAM J. Sci. Comput., 28 (2006), pp. 1798–1811.

[NEARCIRC] L. N. Trefethen, *Near-circularity of the error curve in complex Chebyshev approximation,* J. Approx. Th., 31 (1981), pp. 344–367.

[RATCF] L. N. Trefethen, *Rational Chebyshev approximation on the unit disk,* Numer. Math., 37 (1981), pp. 297–320.

[TDA] L. N. Trefethen, *Ten digit algorithms,* unpublished manuscript, 2005.

[GAUSSCC] L. N. Trefethen, *Is Gauss quadrature better than Clenshaw–Curtis?,* SIAM Rev., 50 (2008), pp. 67–87.

[ATAP] L. N. Trefethen, *Approximation Theory and Approximation Practice, Extended Edition,* SIAM, 2019.

[SERIES] L. N. Trefethen, *Series solution of Laplace problems,* ANZIAM J., 60 (2018), pp. 1–26.

[RATCONFORMAL] L. N. Trefethen, *Numerical conformal mapping with rational functions,*

Computational Methods and Function Theory, submitted, 2019.

[tws] L. N. Trefethen, J. A. C. Weideman, and T. Schmelzer, *Talbot quadratures and rational approximations,* BIT Numer. Math., 46 (2006), pp. 653–670.

[trap] L. N. Trefethen and J. A. C. Weideman, *The exponentially convergent trapezoidal rule,* SIAM Rev., 56 (2014), pp. 385–458.

References

[1] B. Adcock and D. Huybrechs, *Frames and numerical approximation,* SIAM Rev., 61 (2019), pp. 443–473.

[2] A. C. Antoulas and B. D. Q. Anderson, *On the scalar rational interpolation problem,* IMA J. Math. Control Inform., 3 (1986), pp. 61–88.

[3] S. Axler, *Harmonic functions from a complex anlaysis viewpoint,* Amer. Math. Monthly, 39 (1986), pp. 246–258.

[4] G. A. Baker, Jr. and P. Graves-Morris, *Padé Approximants,* Cambridge, 1996.

[5] A. H. Barnett and T. Betcke, *Stability and convergence of the method of fundamental solutions for Helmholtz problems on analytic domains,* J. Comput. Phys., 227 (2008), pp. 7003–7026.

[6] S. N. Bernstein, *Sur l'ordre de la meilleure approximation des fonctions continues par des polynômes de degré donné,* Mém. Acad. Roy. Belg., 1912, pp. 1–104.

[7] F. Bornemann, *Accuracy and stability of computing high-order derivatives of analytic functions by Cauchy integrals,* Found. Comput. Math., 11 (2011), pp. 1–63.

[8] J. P. Boyd, *Solving Transcendental Equations: The Chebyshev Polynomial Proxy and Other Numerical Rootfinders, Perturbation Series, and Oracles,* SIAM, 2014.

[9] C. Brezinski and M. Redivo Zaglia, *Extrapolation Methods: Theory and Practice,* Elsevier, 2013.

[10] R. M. Corless and S. M. Watt, *Bernstein bases are optimal, but, sometimes, Lagrange bases are better,* Proc. SYNASC, Mirton Press, 2004, pp. 141–153.

[11] S. Ellacott and J. Williams, *Rational Chebyshev approximation in the complex plane,* SIAM J. Numer. Anal., 13 (1976), pp. 310–323.

[12] M. D. Finn, S. M. Cox, and H. M. Byrne, *Topological chaos in inviscid and viscous mixers,* J. Fluid Mech., 493 (2003), pp. 345–361.

[13] D. Crowdy, *The Schwarz–Christoffel mapping to bounded multiply connected polygonal domains,* Proc. Roy. Soc. A, 461 (2005), pp. 2653–2678.

[14] P. J. Davis, On the numerical integration of periodic analytic functions, in R. E. Langer, ed., *On Numerical Integration: Proceedings of a Symposium, Madison, April* 21–23*,* 1958*,* Math. Res. Ctr., U. of Wisconsin, 1959, pp. 45–59.

[15] L. M. Delves and J. N. Lyness, *A numerical method for locating the zeros of an analytic function,* Math. Comput., 21 (1967), pp. 543–560.

[16] T. K. DeLillo, A. R. Elcrat, E. H. Kropf, and J. A. Pfaltzgraff, *Efficient calculation of Schwarz–Christoffel transformations for multiply connected domains using Laurent series,* Comput. Meth. Funct. Th., 13 (2013), pp. 307–336.

[17] T. A. Driscoll, *Algorithm 756: A MATLAB toolbox for Schwarz–Christoffel mapping,* ACM Trans. Math. Softw., 22 (1996), pp. 168–186; see also `www.math.udel.edu/~driscoll/SC/`.

[18] G. Fairweather and A. Karageorghis, *The method of fundamental solutions for elliptic boundary value problems,* Adv. Comput. Math., 9 (1998), pp. 69–95.

[19] B. Fornberg, *Numerical differentiation of analytic functions,* ACM Trans. Math. Softw., 7 (1981), pp. 512–526.

[20] B. Fornberg and G. Wright, *Stable computation of multiquadric interpolants for all values of the shape parameter,* Comput. Math. Appl., 48 (2004), pp. 853–867.

[21] D. Gaier, *Lectures on Complex Approximation,* Birkhäuser, 1987.

[22] S. Goedecker, Linear scaling electronic structure methods, Rev. Modern Phys., 71 (1999), pp. 1085–123.

[23] I. J. Good, *The colleague matrix, a Chebyshev analogue of the companion matrix,* Quart. J. Math., 12 (1961), pp. 61–68.

[24] J. Helsing and R. Ojala, *On the evaluation of layer potentials close to their sources,* J. Comput. Phys., 227 (2008), pp. 2899–2921.

[25] P. Henrici, *Fast Fourier methods in complex analysis,* SIAM Rev., 21 (1979), pp. 481–527.

[26] P. Henrici, *Applied and Computational Complex Analysis, v.* 3, Wiley, 1974.

[27] N. J. Higham, *Functions of Matrices: Theory and Computation,* SIAM, 2008.

[28] N. I. Ioakimidis, K. E. Papadakis, and E. A. Perdios, *Numerical evaluation of analytic functions by Cauchy's theorem,* BIT Numer. Math., 31 (1991), pp. 276–285.

[29] R. Jentzsch, *Untersuchungen zur Theorie Analytischer Funktionen,* dissertation, Berlin, 1914.

[30] T. Kato, *Perturbation Theory for Linear Operators, 2nd ed.,* Springer, 1976.

[31] A. Klöckner, A. Barnett, L. Greengard, and M. O'Neil, *Quadrature by expansion: a new method for the evaluation of layer potentials,* J. Comput. Phys., 252 (2013), pp. 332–349.

[32] P. Kravanja and M. Van Barel, *Computing the Zeros of Analytic Functions,* Lect. Notes Math. 1727, Springer, 2000.

[33] C. L. Lawson, *Contributions to the Theory of Linear Least Maximum Approximations,* PhD thesis, UCLA, 1961.

[34] E. Levin and E. B. Saff, *Potential theoretic tools in polynomial and rational approximation,* in Harmonic Analysis and Rational Approximation, Springer, 2006, pp. 71–94.

[35] L. Lin, J. Lu, L. Ying, and W. E, *Pole-based approximation of the Fermi–Dirac function,* Chin. Ann. Math., 30B (2009), pp. 729–742.

[36] R. Luck and J. W. Stevens, *Explicit solutions for transcendental equations,* SIAM Rev., 44 (2002), pp. 227–233.

[37] J. N. Lyness, *Numerical algorithms based on the theory of complex variable,* Proceedings of the 1967 22nd National Conference (1967), pp. 125–133.

[38] J. N. Lyness and C. B. Moler, *Numerical differentiation of analytic functions,* SIAM J. Numer. Anal., 4 (1967), pp. 202–210.

[39] J. E. McCune, *Exact inversion of dispersion relations,* Phys. Fluids, 9 (1966), pp. 2082–2084.

[40] M. M. S. Nasser, *PlgCirMap: A MATLAB toolbox for computing the conformal mapping from polygonal multiply connected domains onto circular domains,* arXiv:1911.01787, 2019.

[41] D. J. Newman, *Rational approximation to $|x|$,* Mich. Math. J., 11 (1964), pp. 11–14.

[42] E. Polizzi, *Density-matrix-based algorithm for solving eigenvalue problems,* Phys. Rev. B, 79 (2009), pp. 115112:1–6.

[43] J. R. Rice and K. H. Usow, *The Lawson algorithm and extensions,* Math. Comput., 22 (1968), pp. 118–127.

[44] C. Runge, *Theorie und Praxis der Reihen,* Göschen, Leipzig, 1904.

[45] E. B. Saff and V. Totik, *Logarithmic Potentials with External Fields,* Springer, 1997.

[46] T. Sakurai and H. Sugiura, *A projection method for generalized eigenvalue problems using numerical integration,* J. Comput. Appl. Math, 159 (2003), pp. 119–128.

[47] H. E. Salzer, *Lagrangian interpolation at the Chebyshev points $x_\nu = \cos(\nu\pi/n)$, $\nu = 0(1)n$; some unnoted advantages,* Comput. J., 15 (1972), pp. 156–159.

[48] O. Sète and J. Liesen, *Properties and examples of Faber–Walsh polynomials,* Comput. Math. Funct. Th., 17 (2017), pp. 151–177.

[49] W. Squire and G. Trapp, *Using complex variables to estimate derivatives of real functions,* SIAM Rev., 40 (1998), pp. 110–112.

[50] H. Stahl, *The convergence of Padé approximants to functions with branch points,* J. Approx. Th., 91 (1997), 139–204.

[51] S. P. Suetin, *Hermite–Padé polynomials and analytic continuation: new approach and some results,* arXiv:1806.08735v1, 2018.

[52] P. T. P. Tang and E. Polizzi, *FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection,* SIAM J. Matr. Anal. Applics., 35 (2014), pp. 354–390.

[53] J.-P. Thiran and M.-P. Istace, *Optimality and uniqueness conditions in complex rational Chebyshev approximation with examples,* Constr. Approx., 9 (1993), pp. 83–103.

[54] J. L. Walsh, *The analogue for maximally convergent polynomials of Jentzsch's theorem,* Duke Math. J., 26 (1959), pp. 605–616.

[55] J. L. Walsh, *Interpolation and Approximation by Rational Functions in the Complex Domain*, AMS, 1935.

[56] H. Wang and D. Huybrechs, *Fast and accurate computation of Chebyshev coefficients in the complex plane,* IMA J. Numer. Anal., 37 (2016), pp. 1150–1174.

[57] E. Wegert, *Visual Complex Functions,* Springer-Birkhäuser, 2012.

[58] R. Wegmann, *Methods for numerical conformal mapping,* in Handbook of Complex Analysis: Geometric Function Theory, v. 2, R. Kühnau, ed., Elsevier, 2005, pp. 351–477.

[59] J. A. C. Weideman, *Computing the dynamics of complex singularities of nonlinear PDEs,* SIAM J. Appl. Dyn. Syst., 2 (2003), pp. 171–186.